Лабораторна робота №3. Архітектура системи команд

Мета лабораторної роботи наступна:

- ознайомитись із елементами рівня архітектури системи команд;
- отримати практичний досвід декодування машинних команд.

1. Вимоги до програмного і апаратного забезпечення

Для виконання лабораторної роботи діють наступні вимоги щодо програмного та апаратного забезпечення:

1) компілятор мови C — gcc версії 14.2 або вище
2) компілятор мови C++ — g++ версії 14.2 або вище.

2. Загальне завдання та термін виконання

Завдання на лабораторну роботу. Реалізувати програму мовою C або C++, що виконує: зчитування послідовності машинних команд (програми), що визначаються варіантом, з файлу; розбір зчитаних команд і вивід їхнього текстового представлення (асемблерного коду). Крім того необхідно реалізувати модульні тести для реалізованого(-их) модуля(-ів) програми.

Програма має містити наступні компоненти (модулі):

1. Модуль з реалізацією функцій зчитування і аналізу (розбору) машинних команд з текстового файлу.
2. Модуль тестування, що містить тестові утиліти і тести реалізованої програми.

В результаті виконання лабораторної роботи має бути підготовлено **звіт**, який містить:

1) титульний аркуш;
2) загальне завдання лабораторної роботи;
3) варіант і завдання за варіантом (інформація про варіанти наведена в п. 4);
4) лістинг реалізованої програми (усіх модулів);
5) результати тестування програми для декількох наборів тестових даних. Тести мають виконувати перевірку всіх пунктів основного завдання, завдання за варіантом і продемонструвати коректність роботи реалізованої програми.

Граничний **термін виконання** лабораторної роботи визначається викладачем в інформаційному листі, що надсилається на пошту групи та/або в групу в телеграмі разом із

даними методичними вказівками. В разі недотримання граничного терміну виконання лабораторної роботи, максимальна оцінка за роботу буде знижена на 1 бал за кожен тиждень протермінування. Кількість балів за дотримання дедлайну даної лабораторної роботи визначається силабусом (РСО). Ця ж кількість відповідає максимальній кількості балів, на яку може бути знижена оцінка за лабораторну роботу за недотримання дедлайну.

3. Методичні вказівки

В даному розділі наведено методичні вказівки щодо виконання пунктів завдання лабораторної роботи.

3.1 Структура команди

Процесорні команди складаються з коду операції (КОП) (операційна частина) і операндів (адресна частина) (рис. 3.1).

Рисунок 3.1 — Структура команди

В полі *коду операції* кодується номер інструкції (дії), яку має виконати процесор. В *полі операндів* знаходяться 0, 1, 2 і більше операндів інструкції (дії), закодованої в КОП. Операндами можуть бути константи, номери регістрів, адреси, тощо. Розмір всієї команди залежить від розміру обох полів і є сумою цих розмірів. І навпаки: розмір полів залежить від того, який загальний розмір команди допустимий у системі команд, що в свою чергу визначається тим, яка система команд використовується — CISC чи RISC. У випадку CISC розмір команди може бути різний — незалежно від розрядності процесора. У випадку RISC розмір команди відповідатиме/буде кратний розрядності процесора.

В даній лабораторній роботі використовується CISC система команд для 32-розрядного процесора. Тобто, розмір регістрів, довжина шини, і, відповідно, адрес — 32 біти, проте, оскільки це CISC, довжина команди може бути різною. Детальніше див. в п. 3.2.

3.2 Система команд

Система команд, що використовується в даній лабораторній роботі наведена в таблиці 3.1. Система команд є обмеженою, тобто, можливо, не містить всіх команд, які можна було б реалізувати в мікропроцесорі. Проте вона містить достатньо команд, щоб отримати уявлення про те, як може виглядати деяка реальна система команд (наприклад система команд архітектури процесорів Intel).

Особливості цієї архітектури та умовні позначення:

- команди системи команд оперують даними, що знаходяться в одному з 16-ти 32-розрядних регістрів загального призначення (R0-R15), або в оперативній пам’яті;

- прапорці аналогічні прапорцям архітектури Intel;
- <reg> в таблиці 3.1 позначає один з регістрів загального призначення;
- <addr> — віртуальна адреса;
- <shift8> — знакове ціле число зміщення відносно поточної команди;
- <lit8>, <lit16>, <lit32> — знакова константа;
- коди команд в таблиці позначаються шістнадцятковими цифрами і символами /regX, /addr, /shift;
- /reg, /reg1, /reg2, /reg3 — номер регістру, 4 біти;
- /addr — адреса, 32 біти;
- /shift — зсув відносно поточної адреси, 8 біт;
- /lit8, /lit16, /lit32 — знакове число розміром 8, 16 і 32 біти відповідно.

Таблиця 3.1 — Система команд

№	Команда	Код команди (0x)	Опис
1	MOV <reg1>, <reg2>	1A /reg1 /reg2 (приклад: 1A 12)	перемістити значення з регістру <reg1> у регістр <reg2>
2	MOV <reg>, <addr>	1B 0 /reg /addr (приклад: 1B 01 00000042)	перемістити значення з ОП за адресою <addr> у регістр <reg>
3	MOV <addr>, <reg>	1B 1 /reg /addr (приклад: 1B 11 00000042)	перемістити значення з регістру <reg> в ОП за адресою <addr>
4	ADD <reg1>, <reg2>	01 /reg1 /reg2 (приклад: 01 21)	додавання значення з регістру <reg1> до значення з регістру <reg2> і збереження результату в регістрі <reg1>
5	ADD <reg>, <addr>	02 0 /reg /addr (приклад: 02 01 00000042)	додавання значення з регістру <reg> до 4-байтового значення з ОП за адресою <addr> і збереження результату в регістрі <reg>
6	ADD <reg1>, <reg2>, <reg3>	03 0 /reg1 /reg2 /reg3 (приклад: 03 01 23)	додавання значення з регістру <reg2> до значення з регістру <reg3> і збереження результату в регістрі <reg1>
7	ADD <reg1>, <reg2>, <addr>	04 /reg1 /reg2 /addr (приклад: 04 12 00000042)	додавання значення з регістру <reg2> до 4-байтового значення з ОП за адресою <addr> і збереження результату в регістрі <reg1>
8	SUB <reg1>, <reg2>	0A /reg1 /reg2 (приклад: 0A 12)	віднімання від значення в регістрі <reg1> значення з регістру <reg2> і збереження результату в регістрі <reg1>
9	SUB <reg>, <addr>	0B 0 /reg /addr (приклад: 0B 01 00000042)	віднімання від значення в регістрі <reg> 4-байтового значення з ОП за адресою <addr> і збереження

			результату в регістрі <reg>
10	SUB <reg1>, <reg2>, <reg3>	0C 0 /reg1 /reg2 /reg3 (приклад: 0C 01 23)	віднімання від значення в регістрі <reg2> значення з регістру <reg3> і збереження результату в регістрі <reg1>
11	SUB <reg1>, <reg2>, <addr>	0D /reg1 /reg2 /addr (приклад: 0D 12 00000042)	віднімання від значення в регістрі <reg2> 4-байтового значення з ОП за адресою <addr> і збереження результату в регістрі <reg1>
12	MUL <reg1>, <reg2>	20 /reg1 /reg2 (приклад: 20 12)	множення значення з регістру <reg1> на значення з регістру <reg2> і збереження результату в регістрі <reg1>. Множення відбувається без розширення, тобто без збереження старших розрядів, якщо результат має розрядність більшу за 32 біти
13	MUL <reg1>, <reg2>, <reg3>	21 0 /reg1 /reg2 /reg3 (приклад: 21 01 23)	множення значення з регістру <reg2> на значення з регістру <reg3> і збереження результату в регістрах <reg1> (старші біти результату) та <reg2> (молодші біти результату)
14	MUL <reg>, <addr>	22 0 /reg /addr (приклад: 22 01 00000042)	множення значення з регістру <reg> на 4-байтове значення з ОП з адресою <addr> і збереження результату в регістрі <reg>. Множення відбувається без розширення, тобто без збереження старших розрядів, якщо результат має розрядність більшу за 32 біти
15	MUL <reg1>, <reg2>, <addr>	23 /reg1 /reg2 /addr (приклад: 23 12 00000042)	множення значення з регістру <reg2> на 4-байтове значення з ОП за адресою <addr> і збереження результату в регістрах <reg1> (старші біти результату) та <reg2> (молодші біти результату)
16	DIV <reg1>, <reg2>	2A /reg1 /reg2 (приклад: 2A 12)	ділення значення з регістру <reg1> на значення з регістру <reg2> і збереження результату в регістрі <reg1>. Зберігається ціла частина від ділення, а залишок не зберігається
17	DIV <reg1>, <reg2>, <reg3>	2B 0 /reg1 /reg2 /reg3 (приклад: 2B 01 23)	ділення значення з регістру <reg2> на значення з регістру <reg3> і збереження результату в регістрах <reg2> (ціла частина) та <reg1> (остача)
18	DIV <reg>, <addr>	2C 0 /reg /addr (приклад: 2V 01 00000042)	ділення значення в регістрі <reg> на 4-байтове значення з ОП за адресою <addr> і збереження результату в регістрі <reg>. Зберігається ціла частина від ділення, а залишок не зберігається
19	DIV <reg1>, <reg2>,	2D /reg1 /reg2 /addr (приклад: 2D 12 00000042)	ділення значення з регістру <reg2> на 4-байтове значення з ОП за адресою

	<addr>		<addr> і збереження результату в регістрах <reg2> (ціла частина) та <reg1> (остача)
20	JMP <shift>	90 /shift (приклад: 90 FA)	безумовний перехід за 1-байтовим відносним зміщенням <shift>
21	JMP <addr>	91 /addr (приклад: 91 00000042)	безумовний перехід за адресою <addr>
22	JL <shift>	92 /shift (приклад: 92 FA)	перехід за 1-байтовим відносним зміщенням <shift> у випадку, якщо SF ≠ OF
23	JL <addr>	93 /addr (приклад: 93 00000042)	перехід за 4-байтовою адресою <addr> у випадку, якщо SF ≠ OF
24	JG <shift>	94 /shift (приклад: 94 FA)	перехід за 1-байтовим відносним зміщенням <shift> у випадку, якщо ZF = 0 і SF = OF
25	JG <addr>	95 /addr (приклад: 95 00000042)	перехід за 4-байтовою адресою <addr> у випадку, якщо ZF = 0 і SF = OF
26	CMP <reg1>, <reg2>	80 /reg1 /reg2 (приклад: 80 12)	порівняння двох значень і встановлення відповідних прапорців
27	MOV <reg>, <lit8>	1C 0 /reg /lit8 (приклад: 1C 01 33)	переміщення 1-байтового числа у регістра <reg>
28	MOV <reg>, <lit16>	1C 1 /reg /lit16 (приклад: 1C 11 3344)	переміщення 2-байтового числа у регістра <reg>
29	MOV <reg>, <lit32>	1C 2 /reg /lit32 (приклад: 1C 21 33445555)	переміщення 4-байтового числа у регістра <reg>

Як видно з таблиці, код операції може мати різну довжину. Наприклад, частина команд кодуються 8-ма бітами (наприклад, команда 1), а частина — 12-ма (наприклад, команди 2 і 3).

3.3 Вхідні дані

На вхід програми подаються наступні файли:

- текстовий файл, що містить деяку послідовність машинних команд (програму) — *файл програми*.

3.4 Модуль зчитування і аналізу файлу програми

Даний модуль має виконувати кілька функцій:

1. Зчитувати програму з файл програми.
2. Аналізувати послідовність команд з файлу програми, вирізняючи в ній окремі коди команд (КОП та операнди). Кожен розпізнаний код команди має бути виведений в термінал у вигляді самої команди з операндами (див. п. 3.4.1, 3.4.2).

3.4.1 Розпізнавання і вивід команд

Нехай маємо наступний вміст файлу програми (коментарі наведені лише для наочності, в файлі з програмою їх не має бути, або вони мають бути пропущені під час зчитування):

```
1C 00 02            // MOV  R0, 2
1B 01 00 C0 00 A1   // MOV  R1, [0x00C000A1]
01 01               // ADD  R0, R1
91 FF FF FF 00      // JMP  [0xFFFFFF00]
```

В загальному випадку (в реальному житті) вміст файлу програми (вхідного файлу) подається у двійковому вигляді — як послідовність кодів команд (машинних команд) згідно із системою команд, визначеною в п. 3.2 і обмеженою конкретним варіантом. Без пробілів та переносів рядків (тобто, розбиття на команди). Проте, для наочності і спрощення виконання тестування можна замість двійкового представлення використовувати послідовність шістнадцяткових цифр — як наведено вище. Кожна команда написана в окремому рядку та з пробілами також лише для наочності та зручності написання тестів і перевірки роботи програми. При цьому **реалізована програма має опрацьовувати вміст файлу програми “по-байтово”, не розраховуючи на те, що команди будуть записані в окремих рядках або між окремим байтами будуть пробіли**.

При зчитуванні кожної наступної команди, слід вивести у термінал рядок з послідовністю байтів команди у шістнадцятковому вигляді, розділеними пробілами, а також символьне представлення команди — у прикладі вище такий рядок позначається після //. Адреси необхідно виводити в квадратних дужках у шістнадцятковому вигляді.

3.4.2 Приклад роботи

Вивід в термінал під час аналізу файлу програми, наведеної в п. 3.4.1, буде наступний:

```
1C 00 02:
MOV  R0, 2

1B 01 00 C0 00 A1:
MOV  R1, [0x00C000A1]

01 01:
ADD  R0, R1

91 FF FF FF 00:
JMP  [0xFFFFFF00]
```

Адреси виводяться в квадратних дужках в шістнадцятковому вигляді з префіксом 0x. Константи і зміщення — десятковими знаковими числами. Регістри — за ім’ям, що є поєднанням літери R і номеру регістру в десятковому вигляді.

3.5 Модуль тестування

Для виконання автоматизовано тестування роботи програми, варто реалізувати модульні та/або інтеграційні тести. Для цього можна написати свої тестові утиліти, або ж скористатись наявними (стандартними) утилітами у мові програмування або спеціалізованими бібліотеками.

Тестові набори мають «покривати» максимальну кількість можливих ситуацій. Тобто, всі команди, задані варіантом, мабуть бути протестовані.

Для того, щоб при реалізації тестів позбутись необхідності створювати файли з тестовими даними (тестовими вхідними програмами), варто в модулі, що виконує зчитування вхідної програми, додати можливість працювати як з файлами, так і з рядками. В такому разі тести можна буде реалізувати виключно у програмному коді, помістивши тестові набори у рядки, при цьому також маючи можливість обробляти файли. При цьому варто уникати дублювання коду — код аналізатора команд має бути один, а от попередня обробка вхідних даних може відрізнятись (обробка рядка, файлу, тощо.).

Аналогічний підхід варто використати для виведення результату: додавання виводу (текстового декодованого представлення машинних команд) має бути реалізовано уніфіковано — наприклад, з використанням потоку (який може бути рядком, файлом, стандартним потоком виводу у консоль).

4. Варіанти завдань

Номер варіанту обирається згідно зі списком групи з варіантами, який був надісланий на пошти груп на початку семестру. Варіант береться по модулю 15 — якщо варіант 16, тоді виконується варіант 1 з таблиці 3.2, якщо 17 — тоді 2 і т. д.

Для кожного варіанту визначається список команд, які має обробляти розроблена програма. Інші команди не мають оброблятись. Список команд містить номери команд з таблиці 3.1.

Тестові набори необхідно створити самостійно.

Таблиця 3.2 — Варіанти завдань

Варіант	**Список команд**
1	1, 2, 12, 14, 24, 25, 26, 27, 29
2	1, 3, 4, 5, 20, 21, 26, 27, 28
3	1, 2, 3, 8, 9, 22, 23, 26, 27
4	1, 3, 10, 11, 13, 15, 26, 27, 28
5	1, 2, 12, 14, 17, 19, 22, 24, 28
6	2, 3, 6, 7, 8, 9, 26, 28, 29
7	1, 3, 16, 17, 18, 19, 21, 26, 28
8	1, 2, 4, 5, 21, 23, 25, 26, 29
9	1, 3, 10, 11, 21, 24, 25, 28, 29
10	1, 2, 6, 7, 13, 15, 20, 21, 27
11	2, 3, 12, 14, 16, 18, 27, 28, 29
12	1, 2, 3, 4, 5, 24, 25, 26, 28

13	1, 3, 8, 9, 13, 15, 22, 23, 26, 28
14	1, 2, 10, 11, 20, 21, 26, 27, 29
15	1, 3, 6, 7, 12, 14, 25, 26, 28

5. Контрольні питання

1) Що таке адресний простір?
2) З чого складається команда?
3) Що таке CISC та RISC системи команди?
4) Що таке модульне тестування?